Housekeeping

- Your goal today: understand the tradeoff between hard and soft logic
- Notices
  - Handout #3: lab 1, due noon, 9/23
- Readings (skim)
The Project Template

- Pick an application
- Pick a metric of merit
- Study implementation options on the Zedboard
  - a good software implementation must be one option
  - the rest is up to you
- Report findings

Keep in mind, you have optimistically 6 weeks; don’t forget you are taking other courses

Degrees of Freedom: you pick the application

- The problem could be
  - well studied (I expect thoroughness and depth)
  - never tried (I give credit for honest attempts)
    Convince us it is 6 weeks of effort
- Something there is a reason to do on FPGAs
  - if you don’t want to think too hard, start from MEMOCODE Design Contests (2007~2016)
  - go thorough FCCM conference proceedings
- Best if it is something you want to or have to do anyways
DoF: you define the metric

• What you can study
  – performance (throughput or latency?)
  – cost (in terms of what?)
  – power and energy (how will you measure?)
  – design effort (what will you measure)
  – app-specific metrics (e.g., numerical accuracy)
  – composite metric: energy-delay-product,
    performance/watt, performance/$

• Defining the metric also means fixing the measurement procedure/protocol

DoF: Platform

• You have the Zedboard
• You may substitute a reconfigurable platform you are already using (check with me first)
• You have access to more advanced platforms (http://users.ece.utexas.edu/~derek/FABRIC.html)
  – risky learning curve to fit in 6 weeks
  – only if this plays into what else you are doing in life
DoF: Approach

• 1 option must be a good software-only baseline
• This is a “study”
  – do more than crank out implementations
  – think about what are the design choices
  – hypothesize the expected effects of your choices
  – corroborate hypothesis by implementation and evaluation

• Implementation approach:
  – no artificial bounds
  – how would you work in real-life?
  – if you have access, you can use it (including tools and IPs)

Convince us it is 6 weeks of effort

What makes a good project

• Interesting and/or important
• Not totally obvious
  (good enough if not obvious to you)
• You have special insights or interest
• Hard enough for 6 weeks
• Not too hard, too risky
• Most importantly, you should enjoy it

The above need not be an AND.
We now return you to our regularly scheduled program . . .

FPGA Pro’s and Con’s

- Reasons for FPGAs
  - no manufacturing NRE (non-recurring eng.) cost
  - faster design time: try out increments as you go
  - less validation time: debug as you go at full speed / can also patch after shipping

- The price of FPGAs
  - high unit cost (not for high-volume products)
  - “~10x” overhead in area/speed/power/….  
  - RTL-level design abstraction (relative to SW)
Hard vs Soft Processor Cores

• Table 4.2: The Zynq Book

<table>
<thead>
<tr>
<th>Processor</th>
<th>Configuration</th>
<th>DMIPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MicroBlaze 900LUT/700FF/2BRAM to 3800LUT/3200FF/6DSP/21BRAM</td>
<td>area optimized (3-stage)</td>
<td>196</td>
</tr>
<tr>
<td></td>
<td>perf. optimized (5-stage) with branch optimizations</td>
<td>228</td>
</tr>
<tr>
<td></td>
<td>perf. optimized (5-stage) without branch optimizations</td>
<td>259 ??from book</td>
</tr>
<tr>
<td>ARM Cortex-A9</td>
<td>1GHz; both cores combined</td>
<td>5000</td>
</tr>
</tbody>
</table>

• Table 4.3: The Zynq Book

<table>
<thead>
<tr>
<th>Processor</th>
<th>Configuration</th>
<th>CoreMark</th>
</tr>
</thead>
<tbody>
<tr>
<td>MicroBlaze</td>
<td>125MHz; 5-stage (Virtex-5)</td>
<td>238</td>
</tr>
<tr>
<td>ARM Cortex-A9</td>
<td>1GHz; both cores combined</td>
<td>5927</td>
</tr>
<tr>
<td>ARM Cortex-A9</td>
<td>800MHz; both cores combined</td>
<td>4737</td>
</tr>
</tbody>
</table>

[Kuon and Rose, 2006]

• Altera Stratix II FPGA, 90nm
  – Quartus II “balanced”, “standard fit”
  – hard multipliers and memory blocks
• ST Micro 90nm standard cells
  – Synopsys “high-effort”, add scan chain
  – ST Micro memory compiler
  – Cadence place and route
• Basic Results
  – avg 21x/40x in area (w/wo using hard macros)
  – 3~4x critical path
  – ~12x dynamic power
Benchmarking

- Opencores and local designs
  - removed cases where FPGA and ASIC are more than 5% different in FF count (Bias?)
- Metrics evaluated
  - logic density
  - circuit speed
  - power consumption

[Table 1: Kuon and Rose, “Measuring the Gap between FPGAs and ASICs,” 2006]

Area Ratios

[Table 2: Kuon and Rose, “Measuring the Gap between FPGAs and ASICs,” 2006]
Critical Path Ratios

Dynamic Power Ratios

[Table 3&4: Kuon and Rose, “Measuring the Gap between FPGAs and ASICs,” 2006]

[Table 5: Kuon and Rose, “Measuring the Gap between FPGAs and ASICs,” 2006]
Actual Mileage Varies

- Comparisons strongly affected by
  - exact design, FPGA/ASIC target, methodology
  - comparing less than “best-effort” designs can bias in either direction
  - design is not a point---a full comparison would have to be Pareto-front to Pareto-front
- Either
  - precise in a specific context, or
  - warm-fuzzy rule of thumb (~10x all around)

\[ 2^{x < "10x" < 100x} \]

Effects of Tuning

- FPGA design not a scaled version of ASIC design
  - different relative cost in logic vs. wires vs. mem
  - different relative speed in logic vs. wires vs. mem
  - also unique usage and operating characteristics

Designed-for-FPGA RTLs need different tuning
FPGA Wire Peculiarities

- Routing architecture over-provisioned to handle worst case
- In a “typical” design, wires appear cheaper relative to other resource types
  
  Best case is nearest-neighbor, regular grid
- Counterintuitively, you SHOULD use wider busses
  - consume unused “free” wires
  - compensate for lower frequency

FPGA Memory Peculiarities

- Large memory abnormally fast
- Large memory are “free” until your run-out
- Quantized memory options
  - jumps between FF-based vs. LUT-RAM vs. BRAMs
  - choose from fixed menu of sizes and aspect ratios
- Must manage RAM usage
  - don’t waste BRAM on small buffers
  - tune buffer sizes to natural granularities, e.g., zero incremental cost to go from 2Kb to 4Kb
  - pack buffers to share same physical array
FPGA Logic Peculiarities

- Logic slower than expected
- Sharp aberrations around hard macro use
e.g., faster mult than add in Virtex-II
- CLB-mapped logic not divisible for pipelining
  - over-pipeline adds cycles without freq. increase
  - “sweetspot” frequency that is easy to reach but hard to exceed
- Design for performance
  - correct and maximal usage of hard macros
  - shallowly pipelined, wide datapath

E.g., FPGA- vs ASIC-tuned NoC on FPGA

- **ASIC RTL** from nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/Router
- **FPGA RTL** from www.ece.cmu.edu/calcm/connect/

![Network Performance](chart)

*Network Performance (uniform random traffic @ 100MHz)*

- Avg. Packet Latency (in ns)
- Load (in Gbps)

*Papamichael, ISFPGA 2012*
Soft-IPs need not be general purpose

- Reconfigurable fabric provides generality
- Soft-IPs should be maximally specialized to usage

<table>
<thead>
<tr>
<th></th>
<th>Uniform Random Traffic</th>
<th>90% Neighbor Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load (in flits/cycle)</td>
<td>Latency (in cycles)</td>
<td>Latency (in cycles)</td>
</tr>
<tr>
<td>0.25</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>0.50</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>0.75</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

It is not just FPGA vs ASIC

- **CPU**: highest-level abstraction / most general-purpose support
- **Multicore**: still high-level abstraction / general parallelism
- **GPU**: explicitly parallel programs / best for SIMD, regular
- **FPGA**: ASIC-like abstraction / overhead for reprogrammability
- **ASIC**: lowest-level abstraction / fixed application and tuning
### Case Study [Chung, MICRO 2010]

<table>
<thead>
<tr>
<th>Device</th>
<th>CPU</th>
<th>GPUs</th>
<th>FPGA</th>
<th>ASIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core i7-960</td>
<td>Nvidia GTX285</td>
<td>ATI RS870</td>
<td>Xilinx V6-LX760</td>
<td>Std. Cell</td>
</tr>
<tr>
<td>Node</td>
<td>45nm</td>
<td>55nm</td>
<td>40nm</td>
<td>40nm</td>
</tr>
<tr>
<td>Die area</td>
<td>263mm²</td>
<td>470mm²</td>
<td>334mm²</td>
<td>-</td>
</tr>
<tr>
<td>Clock rate</td>
<td>3.2GHz</td>
<td>1.5GHz</td>
<td>1.5GHz</td>
<td>0.3GHz</td>
</tr>
</tbody>
</table>

#### Single-prec floating-point apps

<table>
<thead>
<tr>
<th>App</th>
<th>M-M-Mult</th>
<th>FFT</th>
<th>Black-Scholes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MKL 10.2.3</td>
<td>Spiral.net</td>
<td>PARSEC</td>
</tr>
<tr>
<td></td>
<td>Multithreaded</td>
<td>Multithreaded</td>
<td>multithreaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CUBLAS 2.3</td>
<td>CUFFT 2.3</td>
<td>CUDA 2.3</td>
</tr>
<tr>
<td></td>
<td>CAL++</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>hand-coded</td>
<td>Spiral.net</td>
<td>hand-coded</td>
</tr>
</tbody>
</table>

### “Best-Case” Performance and Energy

<table>
<thead>
<tr>
<th>Device</th>
<th>GFLOP/s actual</th>
<th>(GFLOP/s)/mm² normalized to 40nm</th>
<th>GFLOP/J normalized to 40nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core i7 (45nm)</td>
<td>96</td>
<td>0.50</td>
<td>1.14</td>
</tr>
<tr>
<td>Nvidia GTX285 (55nm)</td>
<td>425</td>
<td>2.40</td>
<td>6.78</td>
</tr>
<tr>
<td>ATI RS870 (40nm)</td>
<td>1491</td>
<td>5.95</td>
<td>9.87</td>
</tr>
<tr>
<td>Xilinx V6-LX760 (40nm)</td>
<td>204</td>
<td>0.53</td>
<td>3.62</td>
</tr>
<tr>
<td>same RTL std cell (65nm)</td>
<td>---</td>
<td>19.28</td>
<td>50.73</td>
</tr>
</tbody>
</table>

- CPU and GPU benchmarking is compute-bound; FPGA and Std Cell effectively compute-bound (no off-chip I/O)
- Power (switching+leakage) measurements isolated the core from the system
- For detail see [Chung, et al. MICRO 2010]
### Less Regular Applications

<table>
<thead>
<tr>
<th></th>
<th>GFLOP/s</th>
<th>(GFLOP/s)/mm$^2$</th>
<th>GFLP/J</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FFT 210</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Core i7 (45nm)</td>
<td>67</td>
<td>0.35</td>
<td>0.71</td>
</tr>
<tr>
<td>Nvidia GTX285 (55nm)</td>
<td>250</td>
<td>1.41</td>
<td>4.2</td>
</tr>
<tr>
<td>ATI R5870 (40nm)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Xilinx V6-LX760 (40nm)</td>
<td>380</td>
<td>0.99</td>
<td>6.5</td>
</tr>
<tr>
<td>same RTL std cell (65nm)</td>
<td>952</td>
<td>239</td>
<td>90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Mopt/s</th>
<th>(Mopt/s)/mm$^2$</th>
<th>Mop/s/J</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Black Scholes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Core i7 (45nm)</td>
<td>487</td>
<td>2.52</td>
<td>4.88</td>
</tr>
<tr>
<td>Nvidia GTX285 (55nm)</td>
<td>10756</td>
<td>60.72</td>
<td>189</td>
</tr>
<tr>
<td>ATI R5870 (40nm)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Xilinx V6-LX760 (40nm)</td>
<td>7800</td>
<td>20.26</td>
<td>138</td>
</tr>
<tr>
<td>same RTL std cell (65nm)</td>
<td>25532</td>
<td>1719</td>
<td>642.5</td>
</tr>
</tbody>
</table>

---

### Tradeoff in Heterogeneity?

- Big Core
- Custom Logic
- GPGPU
- FPGA
Amdahl’s Law on Multicore

- A program is rarely completely parallelizable; let’s say a fraction $f$ is perfectly parallelizable
- Speedup of $n$ cores over sequential

\[
\text{Speedup} = \frac{1}{(1 - f) + \frac{f}{n}}
\]

- for small $f$, die area under-utilized

---

Asymmetric Multicores

- Trade pwr/area-efficient “slow” BCEs for a pwr/area-hungry “fast” core
  - fast core for sequential code
  - slow cores for parallel sections
- [Hill and Marty, 2008]

\[
\text{Speedup} = \frac{1 - f}{\frac{1}{\text{perf}_{\text{seq}}} + \frac{f}{(n - r) + \text{perf}_{\text{seq}}}}
\]

- $r$ = cost of fast core in BCE
- $\text{perf}_{\text{seq}}$ = speedup of fast core over BCE
- solve for optimal die area allocation given $f$
Heterogeneous Multicores
[Chung, et al. MICRO 2010]

Asymmetric Fast Core BCE BCE BCE BCE
BCE Fast Core BCE BCE BCE BCE
BCE BCE BCE BCE BCE

Heterogeneous

Base Core Equivalent

For the sake of analysis, break the area for GPU/FPGA/etc. into units of U-cores that are the same size as BCEs. Each U-core type is characterized by a relative performance $\mu$ and relative power $\phi$ compared to a BCE.

\[
\text{Speedup} = \frac{1}{1 - f} \left( \frac{1}{\text{perf}_{\text{seq}}} + f \left( n - r \right) \right)
\]

[Hill and Marty, 2008] simplified $f$ is fraction parallelizable $n$ is total die area in BCE units $r$ is fast core area in BCE units $\text{perf}_{\text{seq}}(r)$ is fast core perf. relative to BCE

\[
\text{Speedup} = \frac{1}{1 - f} \left( \frac{1}{\text{perf}_{\text{seq}}} + f \left( \mu \times (n - r) \right) \right)
\]

$\phi$ and $\mu$ example values

<table>
<thead>
<tr>
<th></th>
<th>MMM</th>
<th>Black-Scholes</th>
<th>FFT-2$^{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nvidia GTX285</td>
<td>$\phi$</td>
<td>0.74</td>
<td>0.57</td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>3.41</td>
<td>17.0</td>
</tr>
<tr>
<td>Nvidia GTX480</td>
<td>$\phi$</td>
<td>0.77</td>
<td>1.29</td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>1.83</td>
<td></td>
</tr>
<tr>
<td>ATI RS870</td>
<td>$\phi$</td>
<td>1.27</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>8.47</td>
<td></td>
</tr>
<tr>
<td>Xilinx LX760</td>
<td>$\phi$</td>
<td>0.31</td>
<td>5.68</td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>0.75</td>
<td>4.75</td>
</tr>
<tr>
<td>Custom Logic</td>
<td>$\phi$</td>
<td>0.79</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>27.4</td>
<td>482</td>
</tr>
</tbody>
</table>

Nominal BCE based on an Intel Atom in-order processor, 26mm$^2$ in a 45nm process.

On equal area basis, 3.41x performance at 0.74x power relative a BCE.
Modeling Power and Bandwidth Budgets

\[ \text{Speedup} = \frac{1}{\frac{1-f}{\text{perf}_{\text{seq}}} + \frac{f}{\mu \times (n-r)}} \]

- The above is based on area alone
- Power or bandwidth budget limits the usable die area
  - if \( P \) is total power budget expressed as a multiple of a BCE's power, then usable U-core area \( n-r \leq P/\phi \)
  - if \( B \) is total memory bandwidth expressed also as a multiple of BCEs, then usable U-core area \( n-r \leq B/\mu \)

Combine Model with ITRS Trends

<table>
<thead>
<tr>
<th>Year</th>
<th>2011</th>
<th>2013</th>
<th>2016</th>
<th>2019</th>
<th>2022</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>40nm</td>
<td>32nm</td>
<td>22nm</td>
<td>16nm</td>
<td>11nm</td>
</tr>
<tr>
<td>Core die budget (mm(^2))</td>
<td>432</td>
<td>432</td>
<td>432</td>
<td>432</td>
<td>432</td>
</tr>
<tr>
<td>Normalized area (BCE)</td>
<td>19</td>
<td>37</td>
<td>75</td>
<td>149</td>
<td>298 (16x)</td>
</tr>
<tr>
<td>Core power (W)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Bandwidth (GB/s)</td>
<td>180</td>
<td>198</td>
<td>234</td>
<td>234</td>
<td>252 (1.4x)</td>
</tr>
<tr>
<td>Rel pwr per device</td>
<td>1X</td>
<td>0.75X</td>
<td>0.5X</td>
<td>0.36X</td>
<td>0.25X</td>
</tr>
</tbody>
</table>

- 2011 parameters reflect high-end systems of the day; future parameters extrapolated from ITRS 2009
- 432mm\(^2\) populated by an optimally sized Fast Core and U-cores of choice
Single-Prec. MMMult (f=99%)

Single-Prec. MMMult (f=90%)
Single-Prec. MMMult (f=50%)

![Graph showing speedup for MMMult with f=50%](image1)

Single-Prec. FFT-1024 (f=99%)

![Graph showing speedup for FFT-1024 with f=99%](image2)
Parting Thoughts

- FPGAs pay an overhead for reconfigurability
  - significant but reducing
  - power and BW bottleneck can compress differences
- FPGAs differ from ASICs in more than then reconfiguration overhead---require distinct architecture and tuning
- {Multicore/GPU/FPGA} are all midpoints between CPU and ASIC extremes
  - none is a panacea
  - go with the easier option unless not good enough